#import necessary packages

import cv2

import os

import numpy as np

from .utils import download\_file

initialize = True

net = None

dest\_dir = os.path.expanduser('~') + os.path.sep + '.cvlib' + os.path.sep + 'object\_detection' + os.path.sep + 'yolo' + os.path.sep + 'yolov3'

classes = None

#colors are BGR instead of RGB in python

COLORS = [0,0,255], [255,0,0]

def populate\_class\_labels():

#we are using a pre existent classifier which is more reliable and more efficient than one

#we could make using only a laptop

#The classifier should be downloaded automatically when you run this script

class\_file\_name = 'yolov3\_classes.txt'

class\_file\_abs\_path = dest\_dir + os.path.sep + class\_file\_name

url = 'https://github.com/Nico31415/Drowning-Detector/raw/master/yolov3.txt'

if not os.path.exists(class\_file\_abs\_path):

download\_file(url=url, file\_name=class\_file\_name, dest\_dir=dest\_dir)

f = open(class\_file\_abs\_path, 'r')

classes = [line.strip() for line in f.readlines()]

return classes

def get\_output\_layers(net):

#the number of output layers in a neural network is the number of possible

#things the network can detect, such as a person, a dog, a tie, a phone...

layer\_names = net.getLayerNames()

output\_layers = [layer\_names[i - 1] for i in net.getUnconnectedOutLayers()]

#output\_layers = [layer\_names[i - 1] for i in net.getUnconnectedOutLayers()]

return output\_layers

def draw\_bbox(img, bbox, labels, confidence, Drowning, write\_conf=False):

global COLORS

global classes

if classes is None:

classes = populate\_class\_labels()

for i, label in enumerate(labels):

#if the person is drowning, the box will be drawn red instead of blue

if label == 'person' and Drowning:

color = COLORS[0]

label = 'DROWNING'

else:

color = COLORS[1]

if write\_conf:

label += ' ' + str(format(confidence[i] \* 100, '.2f')) + '%'

#you only need to points (the opposite corners) to draw a rectangle. These points

#are stored in the variable bbox

cv2.rectangle(img, (bbox[i][0],bbox[i][1]), (bbox[i][2],bbox[i][3]), color, 2)

cv2.putText(img, label, (bbox[i][0],bbox[i][1]-10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, color, 2)

return img

def detect\_common\_objects(image, confidence=0.5, nms\_thresh=0.3):

Height, Width = image.shape[:2]

scale = 0.00392

global classes

global dest\_dir

#all the weights and the neural network algorithm are already preconfigured

#as we are using YOLO

#this part of the script just downloads the YOLO files

config\_file\_name = 'yolov3.cfg'

config\_file\_abs\_path = dest\_dir + os.path.sep + config\_file\_name

weights\_file\_name = 'yolov3.weights'

weights\_file\_abs\_path = dest\_dir + os.path.sep + weights\_file\_name

url = 'https://github.com/Nico31415/Drowning-Detector/raw/master/yolov3.cfg'

if not os.path.exists(config\_file\_abs\_path):

download\_file(url=url, file\_name=config\_file\_name, dest\_dir=dest\_dir)

url = 'https://pjreddie.com/media/files/yolov3.weights'

if not os.path.exists(weights\_file\_abs\_path):

download\_file(url=url, file\_name=weights\_file\_name, dest\_dir=dest\_dir)

global initialize

global net

if initialize:

classes = populate\_class\_labels()

net = cv2.dnn.readNet(weights\_file\_abs\_path, config\_file\_abs\_path)

initialize = False

blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), True, crop=False)

net.setInput(blob)

outs = net.forward(get\_output\_layers(net))

class\_ids = []

confidences = []

boxes = []

for out in outs:

for detection in out:

scores = detection[5:]

class\_id = np.argmax(scores)

max\_conf = scores[class\_id]

if max\_conf > confidence:

center\_x = int(detection[0] \* Width)

center\_y = int(detection[1] \* Height)

w = int(detection[2] \* Width)

h = int(detection[3] \* Height)

x = center\_x - w / 2

y = center\_y - h / 2

class\_ids.append(class\_id)

confidences.append(float(max\_conf))

boxes.append([x, y, w, h])

indices = cv2.dnn.NMSBoxes(boxes, confidences, confidence, nms\_thresh)

bbox = []

label = []

conf = []

for i in indices:

i = i

box = boxes[i]

x = box[0]

y = box[1]

w = box[2]

h = box[3]

bbox.append([round(x), round(y), round(x+w), round(y+h)])

label.append(str(classes[class\_ids[i]]))

conf.append(confidences[i])

return bbox, label, conf